文章编号:1001-2486(2011)06-0061-05

延迟驱动的FPGA高扇出信号线快速布线算法*

陈 迅[1,2],张民选[2]
(1. 信息工程大学 电子技术学院,河南 郑州 450004; 2. 国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:采用基本的延迟驱动Pathfinder布线器对FPGA高扇出信号进行布线,大部分时间会用于初始化寻路的优先级队列,而初始化工作主要是将已得到的布线树中的布线资源结点插入优先级队列。但是分析发现,并非所有被插入的资源结点对布线都是有帮助的。因此提出了一种基于树剪枝的优先级队列初始化算法,这种算法对已有的布线树中的资源点进行筛选后再插入优先级队列。实验结果显示该算法能够取得5.23倍的队列初始化时间加速,在不损失算法结果质量的情况下获得1.55倍的布线加速。

关键词:现场可编程逻辑阵列;布线算法;高扇出信号;树剪枝;延迟驱动
中图分类号:TP391.72 **文献标识码**:A

A Fast Timing-driven Routing Algorithm for FPGA High Fan-out Net

CHEN Xun, ZHANG Min-xuan
(1. Institute of Electronic Technology, Information Engineering Univ, Zhengzhou 450004, China;
2. College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: For base timing-driven PathFinder router, it is found that the High-Fan-Out-Net routing spends around half of the time to insert the previous routed routing tree into routing resource node priority queue, but not all the nodes inserted are useful. In light of this finding, we proposed a pruning tree based priority queue initialization algorithm by just inserting the routing resource node which shares the same direction with next routing sink. For the High-Fan-out Net benchmark, results show our algorithm can shorten the initialization time by 5.23 times, and achieve 1.55 times speedup with almost the same quality of result.

Key words: FPGA(Field Programmable Gate Array); routing algorithm; high-fan-out net; pruning tree; timing-driven

现场可编程逻辑器件(FPGA)以其快速的设计实现速度而占据了大量的集成电路(IC)应用市场。FPGA的设计实现速度主要是设计自动化工具运行时间,包括从硬件描述语言到生成最终配置文件。随着FPGA内包含的逻辑器件规模的飞速增长,现如今工业界最大尺寸的FPGA设计的编译时间已经达到数小时或者数天,而这些时间多半用于进行布局和布线。

布线在编译时间中占据了大多时间,因此需要对其进行加速,并行化[1-7]可以被用于对布线算法进行加速。但在这一情况下,对所有的信号线布线进行加速时,每个处理单元的所分配的布线任务量并不平衡,因为在待布线的信号线集中,并不是每个单一信号线的布线任务量是相同的,例如大扇出信号线的布线难度就要大于小扇出信号线。这一情况会限制到并行化算法的加速度。

在本文中,我们从顺序布线算法研究的角度对并行化布线算法进行了研究。从PathFinder算法在文献[8]中被提出来到在VPR[9-10]中被增强,研究者进行了许多对运行时间的优化,包括Swartz[11]和Tessier[12]的研究。但是,他们的研究仅仅针对的是线长驱动的布线算法,并没有对时序驱动的布线算法进行研究,而时序驱动的布线算法在工业界被广泛应用。

我们证实了Swartz[11]的发现,那就是高扇入信号线在所有运行时间中占据了大量的时间,并不与信号线的数量相关。对MCNC测试电路中的10个电路中高扇出信号线的布线时间进行测量。发现占据全部信号线数量0.56%的高扇出信号线的布线时间占据全部布线时间的70%。而这一测试集被广泛应用于FPGA布线算法的测试,能够较准确地反映实际电路特性。

* **收稿日期**:2011-06-20
资助项目:国家"核高基"重大专项"超高性能CPU新型架构研究"资助项目(2011ZX01028-001-001)
作者简介:陈迅(1981—),男,博士生。

与以往线长驱动的 PathFinder 布线算法的加速研究[11-12]不同，本文主要讨论延迟驱动 PathFinder 算法的加速问题，具体来说是 FPGA 中高扇出信号的布线加速问题。我们提出了一种基于树剪枝的优先级队列初始化算法，这种算法对已有的布线树中的资源点进行筛选后再插入优先级队列，主要工作包括：

（1）提出了一种基于树剪枝的优先级队列初始化算法；

（2）采用试验的方法确定了算法中的两个主要参数，在保证布线结果质量的前提下获得了 1.55 倍的布线速度提升。

1 背景介绍

FPGA 布线算法的主要工作是将布线通道和布线开关分配给信号线。算法中，布线体系结构以布线资源图（routing resource graph）的形式存储。在布线资源图中，布线资源结点（rr_node）代表布线通道，输入输出引脚等资源，而资源图中的边则用来表示这些资源结点是如何连接的[9]。

1.1 基本 PathFinder 算法

本文选择公开源代码的 VPR 软件包[9-10]作为基本 PathFinder 算法的实现。图 1 给出了基本 PathFinder 算法。

```
1  While (congestion exists ){
2   for(each net, i){
3    Rip-up routing tree RT (i) and Update affected
4        p(n) values;
5    RT(i)=NetSource(i);
6    for (each sink, j, of net(i) in decreasing Crit (i,j)
7       order){
8     add the RT(i) into PriorityQueue ;
9     while (sink(i,j) not found){
10      Wave expansion to find the trace to sink  j;
11    }
12    for (all nodes n, in path from RT (i) to sink(i, j)){
13      Update p(n);
14      Add n to RT(i);
15    }
16    Update Elmore delay of RT (i);
17   }
18  }
19  Update h(n) for all n ;
20  Perform timing analysis and update Crit  (i, j) for all
21    nets i and sinks j;
22 }
```

图 1 基本 PathFinder 算法

Fig. 1 Baseline PathFinder algorithm

PathFinder 算法在有资源使用冲突时，对所有信号线进行拆线和重布操作。当所有信号线重布完成时，所有布线资源结点的历史成本（historical cost $h(n)$）将会被更新（第 19 行）。而在每个信号线重布时，所释放或占用的布线资源结点的当前成本（present cost $p(n)$）将会被更新（第 3 ~4 以及 13 行）。

在寻找布线路径时，PathFinder 算法维持了一个优先级队列（*PriorityQueue*，第 8 行），该队列中存储了按照 *TotalCost* 排序的布线资源结点。*TotalCost* 可以通过式（1）计算出来。主要包含 *PathCost* 和 *ExpectedCost* 两部分。这两部分经由一个 α 参数加权以期获得最好的 A^* 算法实现。*PathCost* 代表从信号源到当前布线资源结点的总成本，而 *ExpectedCost* 代表从当前布线资源结点到目标端点的期望成本。

$$TotalCost(n) = PathCost(n) + \alpha \cdot ExpectedCost(n,j) \quad (1)$$

每个信号线仅有一个源点（source）和多个汇点（sink），我们将源点和汇点统称为端点（terminal）。当算法对 s 个目标汇点的信号线 i 进行布线时，首先对这 s 个汇点按照延迟的大小 $Crit(i,j)$ 进行排序，之后这些目标端点按照延迟的降序逐个进行布线。在对目标端点 j 进行布线时，首先对布线资源结点优先队列进行初始化（第 8 行），算法挑选出优先级队列中 *TotalCost* 最小的布线资源结点 n，对与 n 有边相连的资源点 m 计算其 $TotalCost(m)$，并将其插入到优先级队列中。这一过程在算法中被称为行波扩展（wave expansion），行波扩展一直持续到目标端点 j 的路径被找到或者优先级队列为空（第 9 行）。当路径被找到时，路径上所有结点的当前成本都将会被更新并被加入布线树 $RT(i)$。在对第一个汇点进行布线时，信号线 i 的源端 $NetSource(i)$ 被用于初始化优先级队列，而对于其他目标端点，之前的布线树被用于初始化优先级队列。对所有的布线资源结点来说，对不同的目标端点进行布线时 *ExpectedCost* 是不同的，因此优先级队列在更换目标端点时需要清空。

1.2 对大扇出信号布线

首先需要指出，在布线时，是对信号线的汇点按照延迟大小进行排序后逐一进行布线的。当对某一汇点进行布线时，优先级队列首先需要被清空，之后再按照当前需求被初始化。但是这一初始化过程对于高扇出信号线的布线是非常大的开销。对算法的进一步分析（图 1 中第 8 行）可以解释这一开销的由来：首先，有多少汇点，在进行布线时算法将被调用多少次，第二，由于用于对优先级队列进行布线的布线资源点是先前布线的汇点的布线树中的节点，而布线树的规模随着布线过程不断增加。因此，从这两个方面来看，基本 PathFinder 算法对 N 个端点的信号线初始化优先

级队列时将呈现出 $O(N^3)$特性。

对基本 PathFinder 算法的高扇入信号线的布线采用 MCNC 中的 10 个电路进行了分析。对于这些测试电路,我们发现,对于那些端点数量小于 40 的信号线的布线时间与其端点数目成正比,而更大扇出信号线的布线时间与其端点数目却不成正比。因此将高扇出信号线的终端阈值定为 40。在表 1 中,给出了完全的布线时间、高扇出信号线的布线时间以及高扇出信号线的优先级队列初始化(initPQ)时间,其中,比例 1 是高扇出信号线初始化优先级队列占全部信号线布线时间的比例,而比例 2 是高扇出信号线初始化优先级队列占高扇出信号线布线时间的比例。从表格结果中我们可以看出,平均 70% 的全部布线时间被花费在高扇出信号线的布线以及 65% 的高扇出信号线的布线时间被花费在优先级队列初始化时间中。

表 1　基本 PathFinder 运行时间分析

Tab. 1　Baseline PathFinder runtime analyze

测试电路	总布线时间 (ms)	高扇出信号线		初始化优先级队列		
		时间 (ms)	比例 (%)	时间 (ms)	比例 1 (%)	比例 2 (%)
alu4	2801	2070	74	1148	41	55
bigkey	4225	2925	69	1816	43	62
clma	18779	13646	73	8638	46	63
dsip	4544	3651	80	2681	59	73
elliptic	7119	4974	70	3204	45	64
ex1010	8762	5970	68	3067	35	51
s298	4247	3756	88	2293	54	61
s38417	10716	6152	57	4393	41	71
s38584	11356	7787	69	6814	60	88
tseng	1130	637	56	373	33	59
平均值			70		46	65

Swartz 在对布通驱动的 PathFinder 算法进行加速时也发现了这一问题[11]。他们提出了一种 binning 算法来解决这一问题。binning 算法的主要思想是只将那些靠近目标端点的布线资源结点加入优先级队列。但是这种算法是为布通驱动的 PathFinder 设计的,算法需要首先将目标端点按离源结点的距离大小进行排序,而在延迟驱动 PathFinder 算法中,目标端点是按照延迟而进行排序的。

2　基于树剪枝算法的优先级队列初始算法

本文提出的树剪枝算法的主要思想是用布线树中部分结点来初始化优先级队列,被选结点到目标端点和从该结点到先前已布的端点的方向是一致的。图 2 给出了树剪枝算法的一个实例,在这一实例下,对一个有 4 个目标端点的信号进行布线。4 个目标端点是按照图示顺序进行布线的,并且目标端点 1 至 3 已经布线。在对目标端点 4 进行布线时,我们的算法只是将布线树至目标端点 3 的分支中的结点加入优先级队列,而在其他分支上的布线资源结点将被忽略。

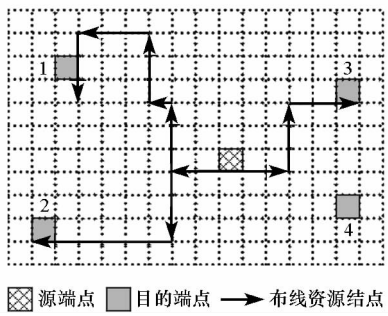

图 2　树剪枝算法实例

Fig. 2　Demonstration of pruning tree algorithm

图 3 给出了树剪枝算法,参数 LEVEL_TH 和 ANGLE_TH 被用于控制布线树的剪枝。

```
1  add_rr_node_to_PriorityQueue ( rr_node n,
2          rr_node target, level l){
3   if (l<=LEVEL_TH){
4    add n into PriorityQueue ;
5    for (all the rr _node m follows n in routing tree ){
6     add_rr_node_to_PriorityQueue(m,
7       target, l+1);
8    }
9   }else{
10   if (is_same_dir(n, target)){
11    add n into PriotityQueuel
12    for (all the rr _node m follows n in routing tree ){
13     add_rr_node_to PriorityQueue(m,
14       target, l+1);
15    }
16   }
17  }
18 }
19 is_same_dir(rr_node n, rr_node target){
20  Set rr_node term be the sink
21      which n is routing to .
22  if (angle(term-n, target-n)<=ANGLE_TH){
23   return TRUE;
24  } else{
25   return FALSE;
26  }
27 }
```

图 3　基于树剪枝的优先级队列初始化算法

Fig. 3　Pruning tree based PriorityQueue initialization algorithm

如果一个布线资源结点 n 在布线树的前 LEVEL_TH 级中,该布线资源结点 n 将被直接加入到优先级队列中(第 4 行),并且结点 n 连接的子树中的布线资源结点将通过递归调用的方式添加入优先级队列(第 5 ~ 8 行);当布线资源结点 n 不在布线树的前 LEVEL_TH 级中,则算法将计算该结点到目标端点和从该结点到先前已布的端点

的夹角并与阈值 ANGLE_TH 进行比较(第 10 行)。如果夹角比阈值小,则说明两方向一致,结点将被加入优先级队列,并且其连接的子树将通过递归调用的方式添加入优先级队列(第 11 ~ 14 行)。

为进一步说明算法如何工作,图 4 给出了一个例子。该信号线有两个目标端点,并且端点 1 已经布线。当对目标端点 2 进行布线时,首先对优先级队列进行初始化,现有布线树将被检查。当布线树中的结点 n_1 和 n_2 被检查时,与目标端点的夹角将被计算,如图中 θ_1 和 θ_2。如果 ANGLE_TH 被设置为 90°,则 n_1 会被添加入优先级队列,而 n_2 将不会。

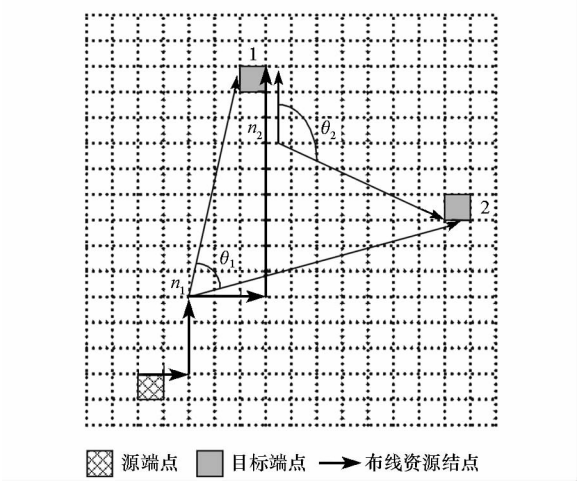

图 4 剪枝实例

Fig. 4 Demonstration of pruning tree

仅仅采用 ANGLE_TH 对布线树进行剪枝并不能获得较好的结果,因为在布线树的顶层几级,算法缺乏每一分支的具体布线方向信息。只有当布线树扩展到下几层,每一分支的具体方向信息已经确定,此时该信息可以被用来裁剪无用的布线资源点。因此,我们增加另一参数 LEVEL_TH 来延迟对树的剪枝。

ANGLE_TH 和 LEVEL_TH 的取值将影响算法的运行时间。大的阈值将减少剪枝数量而增加优先级队列的初始化时间,而小的阈值将增加剪枝数量而增加行波扩展的时间。

3 实验结果

3.1 评价体系

为了比较 VPR PathFinder 和基于树剪枝的 PathFinder 算法,需要对布线时间和布线质量进行比较。

为了比较布线质量,我们对两种算法所需要的最小布线通道值以及关键路径的延迟来进行比较。为了比较布线时间,我们比较所有信号线的布线时间、高扇出信号线的布线时间以及初始化优先级队列的时间。

3.2 实验设置

所有的实验都是基于瓦片式 FPGA 体系结构。相应的体系结构参数如表 2 所示。并且比较关键路径延迟时,采用评估布线算法质量时常用的布线通道宽度设置方法,将其设置为最小通道宽度的 1.5 倍。

表 2 实验所用 FPGA 体系结构参数

Tab. 2 FPGA architecture parameter

参数	数值
LUT 输入数目(K)	4
BLE 中 LUT 中数目(N)	1
开关类型	Wilton
布线通道分段数目($Segment$)	4
接入盒连接参数($f_{C,in}$)	1.0
开关盒连接参数(f_s)	3

此处选择的测试电路来源于 MCNC 测试集,这些测试电路中高扇出信号线的布线时间在所有布线时间中占主要部分。对于剪枝树算法,我们经验性地将两个参数 LEVEL_TH 和 ANGLE_TH 分别设置为 4 和 arctan(10)来获得较好的布线结果。

3.3 实验结果

布线算法布线质量的比较结果如表 3 所示。从表中,我们可以看出,基本 PathFinder 算法和基于树剪枝的 PathFinder 算法的布线质量相当。

表 3 两种算法布线结果比较

Tab. 3 Quality of result comparison of two routing algorithm

测试电路	最小布线通道		关键路径延迟		
	基本算法	树剪枝	基本算法(s)	树剪枝(s)	两者之比
alu4	14	14	4.536e-8	4.536e-8	1.000
bigkey	13	13	4.761e-8	4.581e-8	0.962
clma	16	16	1.253e-7	1.253e-7	0.999
dsip	13	13	3.786e-8	3.710e-8	0.980
elliptic	14	14	9.895e-8	9.907e-8	1.001
ex1010	15	14	1.118e-7	1.122e-7	1.004
s298	11	12	1.071e-7	1.075e-7	1.004
s38417	11	11	9.124e-8	9.041e-8	0.991
s38584	12	12	5.894e-8	5.894e-8	1.000
tseng	9	10	5.654e-8	5.654e-8	1.000
total	128	129			
平均值					0.994

表 4 给出了算法运行时间的对比。从表 3 中,我们可以看出基于树剪枝的 PathFinder 算法在初始化优先级队列时能够取得 5.23 倍的运行时间加速,或者说对高扇出信号线的布线有 2.2 倍的加速,或者说对所有信号线的布线有 1.55 倍的加速。表 4 中同时给出了非高扇出信号线的布线时间来说明布线加速仅是通过高扇出信号线的布线获得的。

表 4　两种算法布线运行时间比较

Tab. 4　Time comparison of two routing algorithm

测试电路	总时间			高扇出信号布线时间			初始化优先级队列时间			其他信号布线		
	基本算法(ms)	裁剪树算法(ms)	加速比	基本算法(ms)	裁剪树算法(ms)	加速比	基本算法(ms)	裁剪树算法(ms)	加速比	基本算法(ms)	裁剪树算法(ms)	加速比
alu4	2800.95	2072.77	1.35	2070.46	1375.49	1.51	1148.39	393.83	2.92	730.49	697.28	1.05
bigkey	4224.51	2330.95	1.81	2925.47	1183.66	2.47	1816.54	279.71	6.49	1299.04	1147.3	1.13
clma	18779.10	14013.25	1.34	13646.77	8265.01	1.65	8638.39	2101.99	4.11	5132.33	5748.24	0.89
dsip	4544.09	2455.18	1.85	3650.72	1497.41	2.44	2681.01	441.93	6.07	893.37	957.72	0.93
elliptic	7118.92	4267.27	1.66	4973.99	2197.21	2.26	3203.51	512.07	6.26	2144.93	2070.05	1.04
ex1010	8762.39	7526.96	1.16	5969.82	4511.66	1.32	3066.83	903.24	3.40	2792.57	3015.30	0.93
s298	4247.36	3322.55	1.28	3756.36	2776.66	1.35	2293.57	730.96	3.14	490.99	545.89	0.90
s38417	10715.68	6434.32	1.67	6151.87	2335.66	2.63	4393.43	707.78	6.21	4563.81	4098.66	1.11
s38584	11356.28	5124.18	2.22	7787.00	1562.36	4.98	6813.77	717.39	9.50	3569.28	3561.82	1.00
tseng	1130.08	971.86	1.16	637.03	449.97	1.42	372.93	87.47	4.26	493.05	521.89	0.94
平均值			1.55			2.20			5.23			0.99

4　结 论

当基本 PathFinder 布线算法被用于对 FPGA 中的高扇出信号进行布线时,大部分时间被花费在将已得到的布线树中的布线资源结点插入优先级队列中,但是并非所有被插入的结点对布线有帮助。提出了一种基于树剪枝的优先级队列初始化算法,该算法只是将那些与目标端点方向一致的布线资源结点插入优先级队列。对于高扇出信号线测试电路集,能够在获得相同的布线质量的前提下获得 1.55 倍的布线速度提升。

参 考 文 献:

[1] Bian H M, Ling A C, Choong A, et al. Towards Scalable Placement for FPGAs [C]//Proceedings of 2010 International Symposium on Field Programmable Gate Arrays, Monterey, California: ACM, 2010:147 - 156.

[2] Ludwin A, Betz V, Padalia K. Highquality, Deterministic Parallel Placement for FPGAs on Commodity Hardware [C]// Proceedings of 2008 International ACM/SIGDA Symposium on Field Programmable Gate Arrays, Monterey, California: ACM, 2008:14 - 23.

[3] Choong A, Beidas R, Zhu J W. Parallelizing Simulated Annealing-based Placement Using GPGPU [C]//2010 International Conference on Field Programmable Logic and Applications, Milano, Italy: IEEE,2010:31 - 34.

[4] Wang C, Lemieux G. Scalable and Deterministic Timing-driven Parallel Placement for FPGAs [C]//Proceedings of 2011 International ACM/SIGDA Symposium on Field Programmable Gate Arrays, Monterey, California: ACM, 2011:153 - 162.

[5] Gort M, Anderson J H. Deterministic Multi-core Parallel Routing for FPGAs [C]//Proceeding of 2010 International Conference on Field-Programmable Technology, Beijing: IEEE, 2010:78 - 86.

[6] Chan P K, et al. Distributed-memory Parallel Routing for Field-programmable Gate Arrays [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2000, 19(8):850 - 862.

[7] Chan P K, Schlag M D F. Acceleration of an FPGA Router [C]//Proceeding of the 5th Annual IEEE Symposium on FPGAs for Custom Computing Machines Napa Valley: ACM, 1997:175 - 181.

[8] Ebeling C, McMurchie L, Hauck S A, et al. Placement and Routing Tools for the Triptych FPGA [J]. IEEE Transactions on Very Large Scale Integration Systems, 1995, 3(4):473 - 482.

[9] Betz V, Rose J. VPR: A New Packing, Placement and Routing Tool for FPGA Research [C]//Proceedings of the 7th International Workshop on Field Programmable Logic and Applications, London:Springer-Verlag, 1997:213 - 222.

[10] Betz V. Architecture and CAD for the Speed and Area Optimization of FPGAs [D]. Toronto:University of Toronto, 1998:229.

[11] Swartz J S, Betz V, Rose J. A Fast Routability-driven Router for FPGAs [C]//Proceedings of the 1998 ACM/SIGDA International Symposium on Field Programmable Gate Arrays. Monterey, California, United States:ACM, 1998:140 - 149.

[12] Tessier R. Negotiated A* Routing for FPGAs [C]// Proceedings of Fifth Canadian Workshop On Field-Programmable Devices , Montreal:IEEE, 1998.